Polytech Dijon

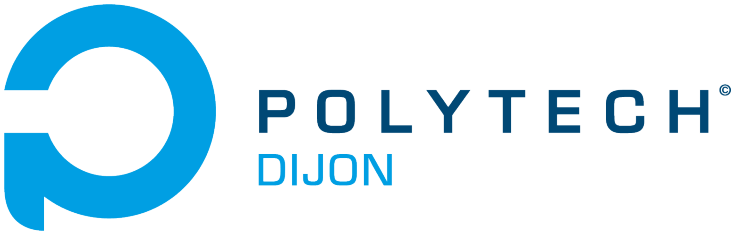
FISA IOT 5A

Compte rendu projet VHDL

Conrolleur SRAM

Auteur :
HELLE Evan

Enseignant :
DUBOIS Julien



2025-2026

Sommaire

1 Compte Rendu **5**
1.1 Introduction 5
1.2 IO Buffer 5
1.2.1 Fonctionnement 5
1.2.2 Test Bench 6
1.3 Crontrolleur SRAM 8
1.3.1 Finished State Machine 8
1.3.2 Test bench 8
1.4 Conclusion 8
1.5 Remerciements 8

2 Annexe **9**
2.1 Code-source / Document... 9
2.2 Bibliographie 9

Table des Figures

1.1 IO buffer 5
1.2 Extrait ennoncé projet 6
1.3 Simulation IO Buffer 8
1.4 Haute impédance à la lecture de la donnée 8

Ce document a été réalisé avec LaTeX.

1 Compte Rendu

1.1 Introduction

Ce projet consiste à concevoir et implémenter en VHDL un contrôleur pour une mémoire SRAM de type mt55l512y36f, dont le modèle VHDL est fourni par le fabricant.
L'objectif est de permettre les opérations de lecture et d'écriture tout en respectant les contraintes temporelles et les spécifications électriques du composant.

Le contrôleur recevra les données via une interface d'entrée-sortie (IO Buffer) et assurera la communication avec la mémoire SRAM à travers une interface dédiée.
Son fonctionnement sera structuré autour d'une machine à états finis (FSM) afin de gérer de manière séquentielle et fiable les différentes phases d'accès mémoire.

Le projet inclura également la réalisation de `test bench` destinés à valider le comportement et la conformité de chaque élément du contrôleur.

1.2 IO Buffer

1.2.1 Fonctionnement

Un IO Buffer est un composant matériel utilisé dans les circuits pour gérer les signaux d'entrée et de sortie entre le circuit et le monde extérieur.
L'implémentation de L'IO Buffer que nous allons utilisé nous nous a été fourni.

Voici la représentation schématique de l'IO Buffer que nous allons utilisé dans notre projet :

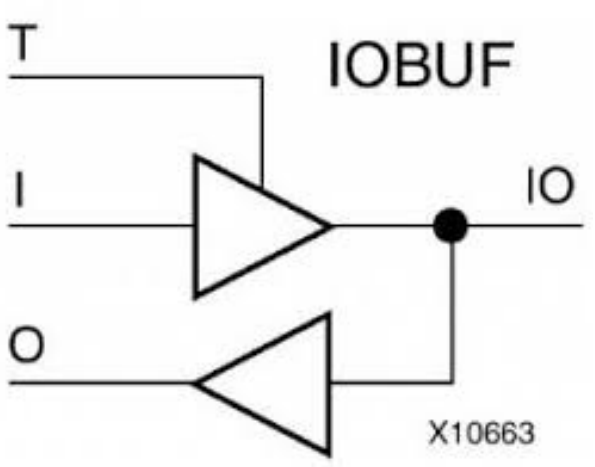


Figure 1.1: IO buffer

L'IO Buffer est composé de plusieurs entrées et sorties :

- **I** : Donnée d'entrée (input data)
- **O** : Donnée de sortie (output data)
- **T** : Trigger, permet de contrôler le mode de l'IO Buffer (entrée ou sortie qui servira à gerer l'ecriture ou la lecture dans notre cas)
- **IO** : Bus de données bidirectionnel (data bus)

1.2.2 Test Bench

L'objectif de faire un test bench est de vérifier le bon fonctionnement de l'IO Buffer avec la SRAM.
De plus cela premettra de de comprendre son fonctionnement avant de l'intégrer dans le projet final.
Il est important de faire ce test car il est nécessaire d'avoir un décalage entre la donnée et les signaux de contrôle au niveau de la SRAM.

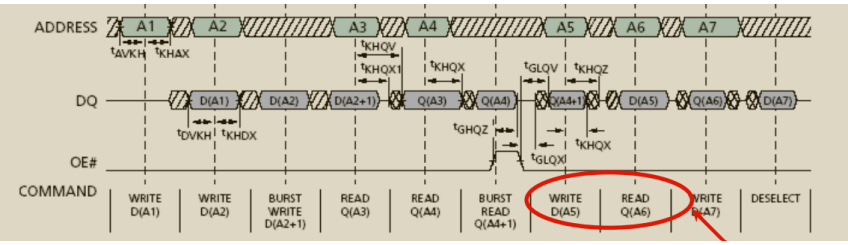


Figure 1.2: Extrait ennoncé projet

D'après l'ennoncé du projet et la datasheet de la SRAM, il est nécessaire d'avoir un décalage d'un cycle d'horloge entre la donnée et les signaux de contrôle.

Code

Pour le code du test bench, je nous sommes partie sur une ecriture de donnée à l'adresse 1 et 2, puis une lecture de ces deux adresses.

```
tb : PROCESS
BEGIN
-- init
nCKE    <= '0';
nADVLD  <= '0';
nOE     <= '0';-- output enable
nCE     <= '0';
nCE2    <= '0';
CE2     <= '1';
SA      <= (others => '0');
Trig    <='1'; -- se mettre en "lecture" le temps de l'init pour ne pas ecrire n'importe quoi

wait for 6*(TCLKL+TCLKH);
```

```
SA                  <= "000"&x"0001";
Trig    <= '0'; -- ecriture à l'adresse 1
wait for 1*(TCLKL); -- pour travailler sur front montant
ENTREE  <= (others => '1'); -- decalage de la donnée d'un cycle
                            --par rapport à l'adresse et la commande


wait for 2*(TCLKL+TCLKH);
SA                  <= "000"&x"0002"; -- eriture à l'adress 2
wait for 1*(TCLKL+TCLKH);
ENTREE  <= ENTREE + 1;-- decalage de la donnée d'un cycle par
                      -- rapport à l'adresse et la commande

wait for 2*(TCLKH+TCLKL);
SA                  <= "000"&x"0001"; -- lecture à l'adresse 1
Trig <= '1';
wait for 2*(TCLKH+TCLKL);
SA                  <= "000"&x"0002"; -- lecture à l'adresse 2

wait; -- will wait forever
END PROCESS;
```

Ici la donnée est bien décalée d'un cycle par rapport à l'adresse et la commande. Quand le trigger est à 0, on est en écriture, et quand il est à 1, on est en lecture. C'est le cas pour la SRAM et l'IO buffer. Donc nous avons connecter par un fil ces deux I/O.

```
SRAM1 : mt551512y36f port map
(DQ, SA, '0', CLKO_SRAM, nCKE, nADVLD, '0',
'0', '0', '0', Trig, nOE, nCE, nCE2, CE2, '0');

IOb: for I in 0 to 35 generate
Iobx: IOBUF_F_16  port map(
O => SORTIE(I),
IO => DQ(I),
I => ENTREE(I),
T => Trig
);
end generate;
```

Simulation

Voici le résultat de la simulation :

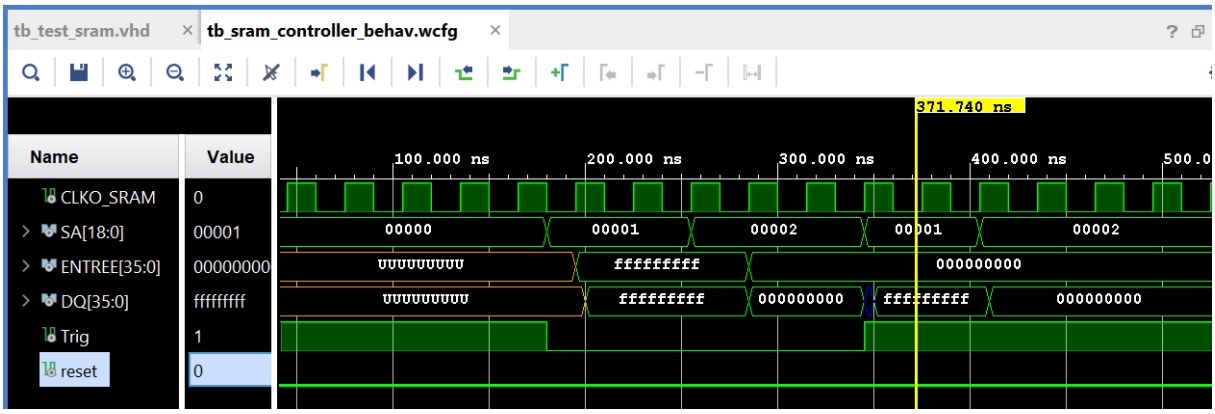


Figure 1.3: Simulation IO Buffer

Les données sont bien écrites aux bonnes adresses (1 et 2) et lues correctement par la suite.
Au passage à la lecture, le bus bidirectionnel passe en haute impédance, ce qui est le comportement attendu.
Voici un zoom sur le passage en haute impédance :

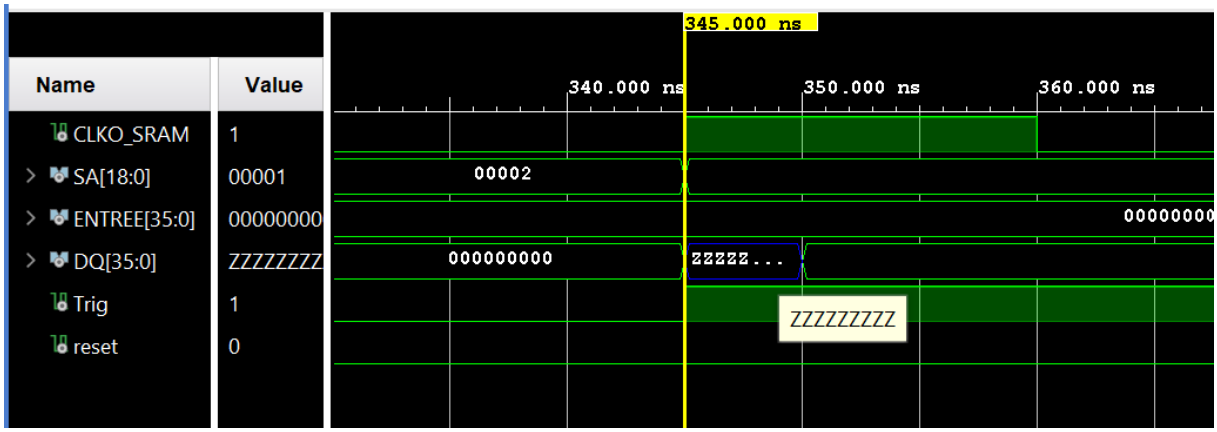


Figure 1.4: Haute impédance à la lecture de la donnée

1.3 Crontrolleur SRAM

1.3.1 Finished State Machine

1.3.2 Test bench

1.4 Conclusion

1.5 Remerciements

2 Annexe

2.1 Code-source / Document...

2.2 Bibliographie